Exam Solution

Ahmed Alhasan

3/24/2020

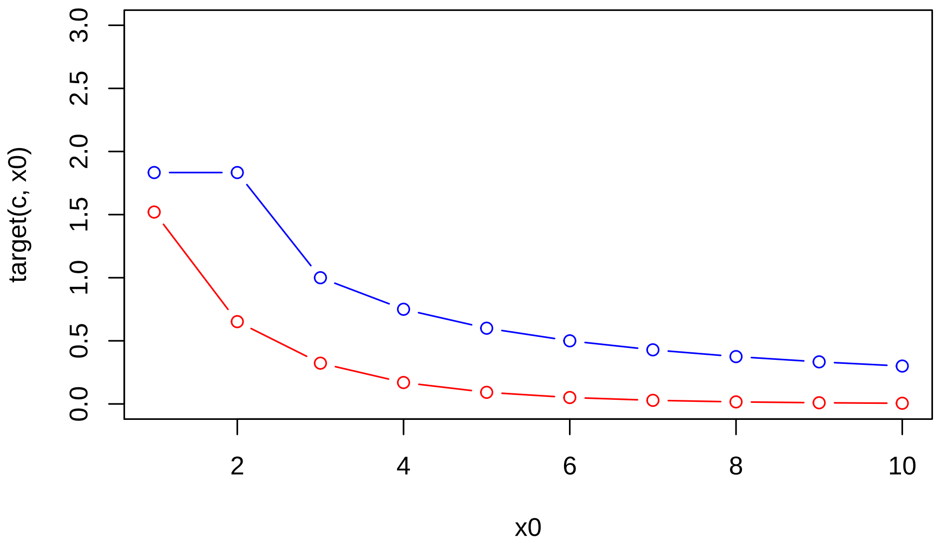
I solemnly swear upon my honour that I wrote the exam honestly, I did not use any unpermitted aids, nor did I communicate with anybody except of the course examiners.

Ahmed Alhasan

Assignment 1

1.1

```r
# Target Distribution
target <- function(c,x){
  c * sqrt(2 * pi) * exp(-c^2/2*x) * x^(-3/2) * x
}
#c <- runif(1,1,10)
c <- 1
x0 <- 1:10
plot(x0, target(c,x0), type = "b", col = "red", ylim = c(0,3))



# Power_Law
p <- function(a,t,x){
  ((a-1) / t) * (x/t)^(-a) * x
}




# Because the parameter T_min controls x we will have problem sampling from the
# region [0,T_min) to avoid this we can use the expected value of p(x) to replace
# the missing values

#a <- runif(1,1.1,10)
a <- 2
#t <- runif(1,0,10)
t <- 3
x <- t:10
maj <- function(a,t,x){
  if(t == 1){
    y <- p(a,t,x)
  }

  else{
    m <- rep(mean(p(a,t,1:t)),length(1:(t-1)))
    y <- append(m,p(a,t,x))
  }
  c <- 1
  x0 <- 1:10
  plot(x0, target(c,x0), type = "b", col = "red", ylim = c(0,3))
  points(x0, y, type = "b", col = "blue")
}
maj(a,t,x)
```



- Because the parameter T_min controls x we will have problem sampling from the region [0,T_min) to avoid this we can use the expected value of p(x) to replace the missing values
- the majorizing constant can be selected as 1 if we choose a = 1 and t = 1, meaning we dont need to multiply by c
- for any value of alpha (1,Inf] and values of T_min[2,Inf] p(x) can work as a majorizing function

1.2

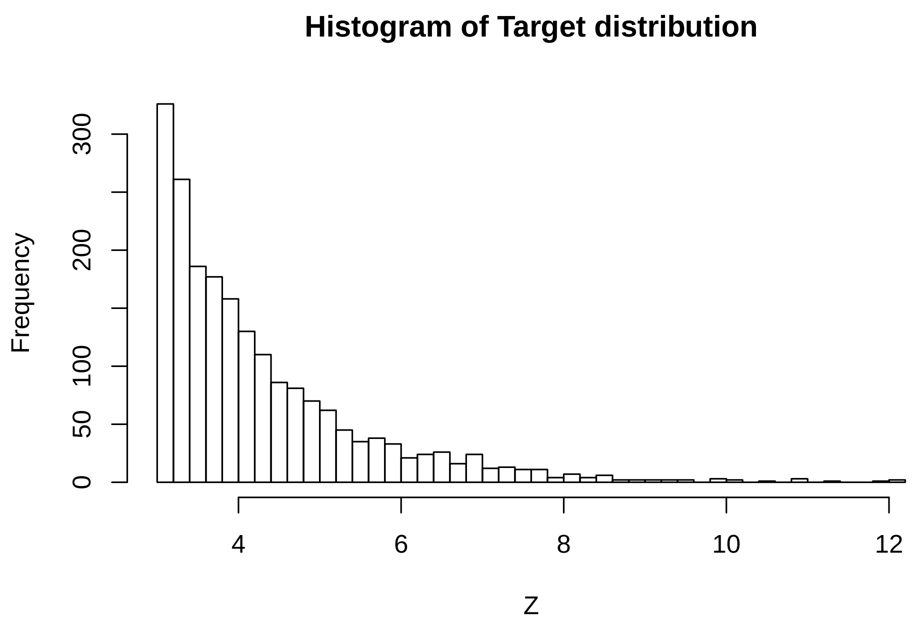
```
library(poweRlaw)
accept_reject <- function(n,c){
  R <- 0
  Y <- vector(length = n)
  for(i in 1:n){
    repeat {
      y <- rplcon(1,3,2)
      U <- runif(1)
      h <- target(1,y) / (c * p(2,3,y))
      if(U <= h){
        Y[i] <- y
        break
      }
      else{R = R+1}
    }
  }
  return(list(Y=Y, Reject=R))
}
```

1.3

```
set.seed(12345)
Z1 <- accept_reject(n = 2000, c = 1)
cat("The number of rejections: ",Z1$Reject)
```

```
## The number of rejections:  14168
```

```
hist(Z1$Y, xlab = "Z", main = "Histogram of Target distribution", breaks = 50)
```



```
cat("Mean: ", mean(Z1$Y))
```

```
## Mean:  4.266329
```

```r
cat("Variance: ", var(Z1$Y))
```

```
## Variance:  1.697752
```

```r
set.seed(12345)
Z2 <- accept_reject(n = 2000, c = 2)

cat("Mean: ", mean(Z2$Y))
```

```
## Mean:  4.290992
```

```r
cat("Variance: ", var(Z2$Y))
```

```
## Variance:  1.917827
```

```r
set.seed(12345)
Z3 <- accept_reject(n = 2000, c = 3)

cat("Mean: ", mean(Z3$Y))
```

```
## Mean:  4.272994
```

```r
cat("Variance: ", var(Z3$Y))
```

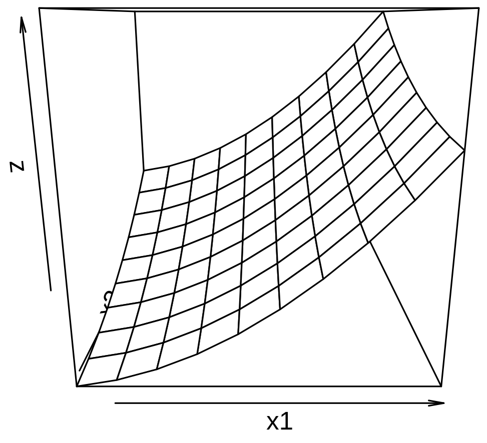
```
## Variance:  1.941761
```

- Mean and Variance do not depend on the value of c since c only controls the majorizing function while the target function is always the same no matter what c value is.

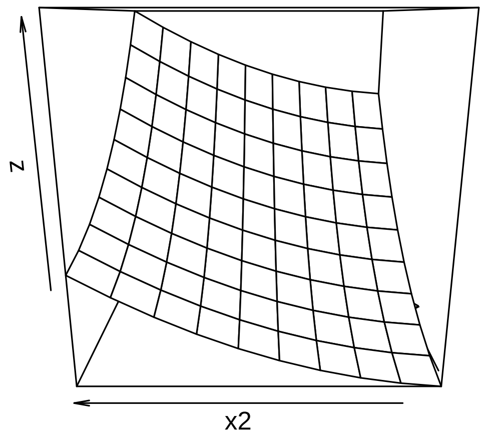
Assignment 2

2.1

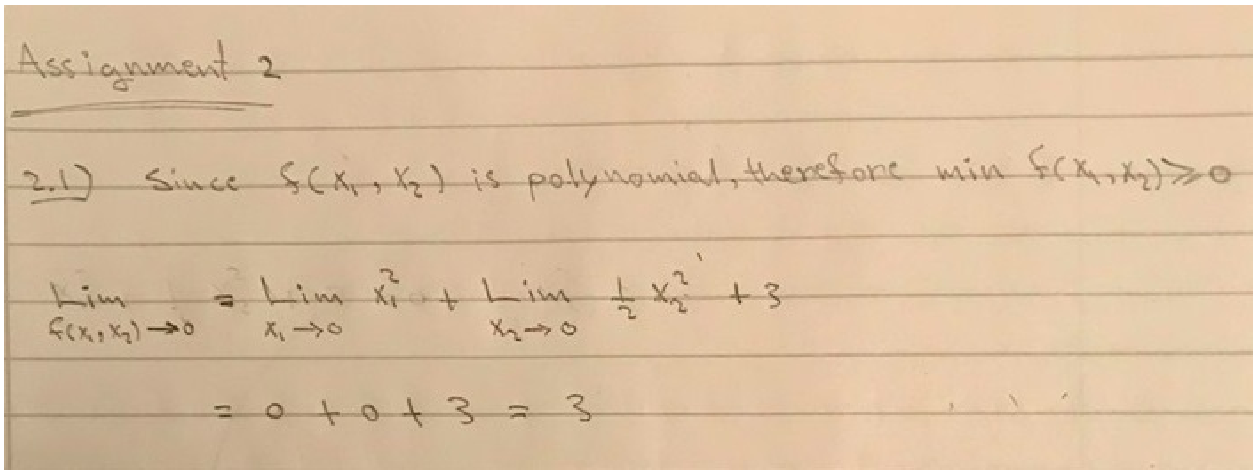
```r
f <- function(x1,x2){
  x1^2 + (1/2) * x2^2 + 3
}

x1 <- 1:10
x2 <- 1:10
z <- outer(x1, x2, f)
persp(x1,x2,z, theta = 0)
```



```
persp(x1,x2,z, theta = -90)
```



```
a1 <- append(rep(0,30), c(1,1))
a2 <- append(rep(0,29), c(1,0,0))
a3 <- append(rep(0,29), c(1,0,1))
a4 <- append(rep(0,29), c(1,1,0))
a5 <- append(rep(0,29), c(1,1,1))
```


Assignment 2

2.1) Since $f(x_1, x_2)$ is polynomial, therefore min $f(x_1,x_2) \geq 0$

$$\lim_{f(x_1,x_2)\to 0} = \lim_{x_1 \to 0} x_1^2 + \lim_{x_2 \to 0} \frac{1}{2} x_2^2 + 3$$

$$= 0 + 0 + 3 = 3$$

Figure 1: minimum

```
a6 <- append(rep(0,28), c(1,0,0,0))
a7 <- append(rep(0,28), c(1,0,0,1))
a8 <- append(rep(0,28), c(1,0,1,0))
a9 <- append(rep(0,28), c(1,0,1,1))
a10 <- append(rep(0,28), c(1,1,0,0))
pop1 <- rbind(a1,a2,a3,a4,a5,a6,a7,a9,a10)
pop2 <- pop1

y1 <- c()
x_1 <- c()
for(i in 1:8){
  y1[i] <- Reduce(function(s,r) {s*2+r}, pop1[i,])
  x_1[i] <- (-1)^(pop1[i,1]) * exp(y1[i] *(-1)^(pop1[i,2]))
}

y2 <- c()
x_2 <- c()
for(i in 1:8){
  y2[i] <- Reduce(function(s,r) {s*2+r}, pop2[i,])
  x_2[i] <- (-1)^(pop2[i,1]) * exp(y2[i] *(-1)^(pop2[i,2]))
}
```

2.2

2.3

```
crossover <- function(x,y){
  (x+y)/2
}

mutate <- function(x){
  x^2 %% 30
}
```

```r
genetic <- function(maxiter, mutprob){
  x1 <- 1:10
  x2 <- 1:10
  Values <- f(x1,x2)
  min_val <- Inf
  for(i in 1:maxiter){
    parents_1 <- sample(x1, size = 2)
    parents_2 <- sample(x2, size = 2)

    victim <- order(Values)[1]

    kid_1 <- crossover(x = parents_1[1], y = parents_1[2])
    kid_2 <- crossover(x = parents_2[1], y = parents_2[2])
    kid_1 <- ifelse(mutprob > runif(1), mutate(kid_1), kid_1)
    kid_2 <- ifelse(mutprob > runif(1), mutate(kid_2), kid_2)
    x1[victim] <- kid_1
    x2[victim] <- kid_2
    Values <- f(x1,x2)
    min_val <- min(min_val, min(Values))
  }
  list(optimum = min_val, population1 = x1, population2 = x2, Values = Values)
}

set.seed(12345)
genetic(maxiter = 100, mutprob = 0.0077)
```

```
## $optimum
## [1] 9
##
## $population1
##  [1]  9.980091  9.950963  9.964635  9.967743  9.953206  9.950163  9.960183
##  [8]  9.961352  9.962993 10.000000
##
## $population2
##  [1]  9.966318  9.966318  9.958175  9.936103  9.957977  9.960161  9.950033
##  [8]  9.964413  9.961154 10.000000
##
## $Values
##  [1] 152.2660 151.6854 151.8766 151.7190 151.6470 151.6082 151.7068 151.8733
##  [9] 151.8735 153.0000
```

```r
set.seed(12345)
genetic(maxiter = 100, mutprob = 0.5)
```

```
## $optimum
## [1] 9
##
## $population1
##  [1] 26.25000 28.43780  7.79896 28.54945 28.79707 28.58918 27.52353 28.62891
##  [9] 28.07853 28.44165
##
## $population2
##  [1] 28.265625 23.084161  8.601327 22.811277 27.373778 21.067123 26.575567
##  [8] 23.931915 25.777357 24.666374
##
## $Values
##  [1] 1091.5353 1078.1478  100.8152 1078.2483 1206.9331 1042.2530 1113.6754
##  [8] 1108.9826 1123.6401 1116.1426
```

Appendix

```
knitr::opts_chunk$set(echo = TRUE, fig.align = "center", out.width = "80%", warning = FALSE)
# Target Distribution
target <- function(c,x){
  c * sqrt(2 * pi) * exp(-c^2/2*x) * x^(-3/2) * x
}
#c <- runif(1,1,10)
c <- 1
x0 <- 1:10
plot(x0, target(c,x0), type = "b", col = "red", ylim = c(0,3))



# Power_Law
p <- function(a,t,x){
  ((a-1) / t) * (x/t)^(-a) * x
}



# Because the parameter T_min controls x we will have problem sampling from the
# region [0,T_min) to avoid this we can use the expected value of p(x) to replace
# the missing values

#a <- runif(1,1.1,10)
a <- 2
#t <- runif(1,0,10)
t <- 3
x <- t:10
maj <- function(a,t,x){
  if(t == 1){
    y <- p(a,t,x)
  }

  else{
    m <- rep(mean(p(a,t,1:t)),length(1:(t-1)))
    y <- append(m,p(a,t,x))
  }
  c <- 1
  x0 <- 1:10
  plot(x0, target(c,x0), type = "b", col = "red", ylim = c(0,3))
  points(x0, y, type = "b", col = "blue")
}
```

```r
maj(a,t,x)

library(poweRlaw)
accept_reject <- function(n,c){
  R <- 0
  Y <- vector(length = n)
  for(i in 1:n){
    repeat {
      y <- rplcon(1,3,2)
      U <- runif(1)
      h <- target(1,y) / (c * p(2,3,y))
      if(U <= h){
        Y[i] <- y
        break
      }
      else{R = R+1}
    }
  }
  return(list(Y=Y, Reject=R))
}
set.seed(12345)
Z1 <- accept_reject(n = 2000, c = 1)
cat("The number of rejections: ",Z1$Reject)

hist(Z1$Y, xlab = "Z", main = "Histogram of Target distribution", breaks = 50)

cat("Mean: ", mean(Z1$Y))
cat("Variance: ", var(Z1$Y))


set.seed(12345)
Z2 <- accept_reject(n = 2000, c = 2)

cat("Mean: ", mean(Z2$Y))
cat("Variance: ", var(Z2$Y))


set.seed(12345)
Z3 <- accept_reject(n = 2000, c = 3)

cat("Mean: ", mean(Z3$Y))
cat("Variance: ", var(Z3$Y))
f <- function(x1,x2){
  x1^2 + (1/2) * x2^2 + 3
}

x1 <- 1:10
x2 <- 1:10
z <- outer(x1, x2, f)
persp(x1,x2,z, theta = 0)
persp(x1,x2,z, theta = -90)
a1 <- append(rep(0,30), c(1,1))
a2 <- append(rep(0,29), c(1,0,0))
```

```
a3 <- append(rep(0,29), c(1,0,1))
a4 <- append(rep(0,29), c(1,1,0))
a5 <- append(rep(0,29), c(1,1,1))
a6 <- append(rep(0,28), c(1,0,0,0))
a7 <- append(rep(0,28), c(1,0,0,1))
a8 <- append(rep(0,28), c(1,0,1,0))
a9 <- append(rep(0,28), c(1,0,1,1))
a10 <- append(rep(0,28), c(1,1,0,0))
pop1 <- rbind(a1,a2,a3,a4,a5,a6,a7,a9,a10)
pop2 <- pop1

y1 <- c()
x_1 <- c()
for(i in 1:8){
  y1[i] <- Reduce(function(s,r) {s*2+r}, pop1[i,])
  x_1[i] <- (-1)^(pop1[i,1]) * exp(y1[i] *(-1)^(pop1[i,2]))
}

y2 <- c()
x_2 <- c()
for(i in 1:8){
  y2[i] <- Reduce(function(s,r) {s*2+r}, pop2[i,])
  x_2[i] <- (-1)^(pop2[i,1]) * exp(y2[i] *(-1)^(pop2[i,2]))
}


crossover <- function(x,y){
  (x+y)/2
}

mutate <- function(x){
  x^2 %% 30
}

genetic <- function(maxiter, mutprob){
  x1 <- 1:10
  x2 <- 1:10
  Values <- f(x1,x2)
  min_val <- Inf
  for(i in 1:maxiter){
    parents_1 <- sample(x1, size = 2)
    parents_2 <- sample(x2, size = 2)

    victim <- order(Values)[1]

    kid_1 <- crossover(x = parents_1[1], y = parents_1[2])
    kid_2 <- crossover(x = parents_2[1], y = parents_2[2])
    kid_1 <- ifelse(mutprob > runif(1), mutate(kid_1), kid_1)
    kid_2 <- ifelse(mutprob > runif(1), mutate(kid_2), kid_2)
    x1[victim] <- kid_1
    x2[victim] <- kid_2
    Values <- f(x1,x2)
    min_val <- min(min_val, min(Values))
```

```r
  }
  list(optimum = min_val, population1 = x1, population2 = x2, Values = Values)
}

set.seed(12345)
genetic(maxiter = 100, mutprob = 0.0077)

set.seed(12345)
genetic(maxiter = 100, mutprob = 0.5)
```